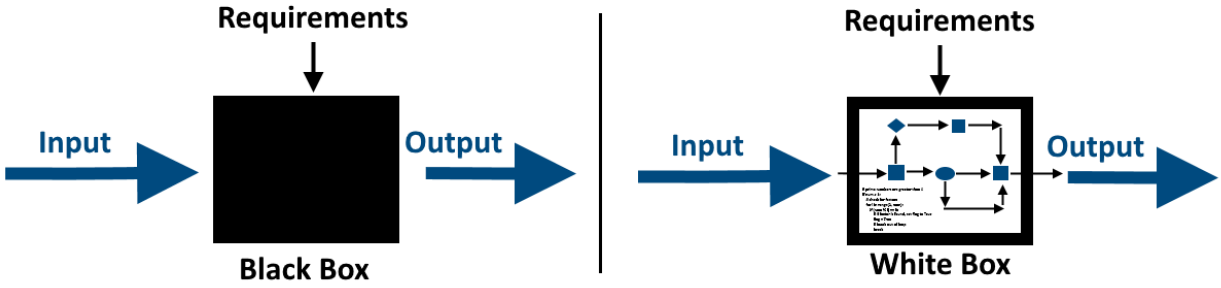
Testiranje belom i crnom kutijom

U uvodnim lekcijama ovog kursa, govorili smo o black box i white box testiranju. U okviru ove lekcije, podsetićemo se šta predstavljaju black box i white box testiranje, sumirati kako smo do sada savladali neke od važnijih principa i predstaviti metode testiranja o kojima ćemo pričati u narednim lekcijama. Ovo će nam poslužiti kao teorijski uvod u dalju automatizaciju testiranja i oblast kontrole kvaliteta.

Ova lekcija ujedno predstavlja i prvu lekciju oblasti kontrole kvaliteta. Sve metode i tehnike koje smo prikazali do sada se bave obezbeđenjem kvaliteta. Dakle, kako da osiguramo da naš program radi ono što je očekivano. Sada dolazimo do sledećeg nivoa – nivoa na kojem rade softver testeri. Kada na višem standardu kontrolišemo sve aspekte, i uspeh ovih vidova testiranja zavisi od toga u kojoj meri je u QA fazi kod programa napisan pravilno i prema specifikaciji zahteva.

Testiranje na nivou sistema

Kada smo govorili o načinu testiranja programa, spomenuli smo upravo black box testiranje ili funkcionalno testiranje, kao i white box testiranje ili strukturalno testiranje.



Slika 10.1. Black box i white box testiranje

Rekli smo da ovi pristupi pripadaju grupi testiranja u zavisnosti od pristupa sistemu.

I iz tih lekcija smo mogli da izvučemo zaključak: black box testiranje obavljamo najčešće nad celim sistemom, bez pristupa kodu i pregleda koda. Dakle, pokrećemo aplikaciju i pratimo njen rad i ponašanje. White box testiranje obavljamo u okviru samog koda, dodavanjem linija i uslova i definisanjem testova koji vrše interakciju direktno sa blokom koda.

Sada kako smo malo više radili sa samim testiranjem i različitim pristupima, možemo pogledati neke razlike između black box i white box testiranja:

Black box testiranje	White box testiranje
Testiranje se vrši bez poznavanja koda ili pristupa kodu	Testiranje se vrši sa pristupom kodu i potrebno je poznavanje koda
Testiranje obavljaju softver testeri, posebni članovi tima specijalizovani za ovaj vid testiranja	Testiranje obavljaju programeri koji su razvili softver
Često se naziva spoljnim testiranjem	Često se naziva internim ili unutrašnjim testiranjem
Predstavlja testiranje funkcionalnosti	Predstavlja testiranje strukture koda
Testira se ponašanje sistema	Testira se logika koda

Tabela 10.1. Poređenje black box i white box testiranja

Takođe, jedna vrlo važna podela kojom smo se vodili do sada jeste podela prema nivou testiranja:

- Testiranje jedinica
  Testiranje jedinica (unit testing) se odnosi na testiranje pojedinačnih jedinica izvornog koda ili delova klase.

- Integraciono testiranje
  Nakon izvršenog testiranja jedinica, ispravne jedinice se integrišu u celine poput paketa i modula. Tada na scenu stupa integraciono testiranje. Glavni fokus ovog vida testiranja je na verifikaciji funkcionalnosti i veza (interfejsa) između integrisanih modula.

- Testiranje sistema
  Na kraju, kada su svi delovi završeni i provereni, započinje se sa testiranjem sistema, gde proveravamo ponašanje sistema kao celine u odnosu na očekivane zahteve koje sistem treba da ispuni. Kako je većina funkcionalnih zahteva proverena u okviru nižih nivoa testova, ovde se akcenat postavlja na nefukcionalne zahteve, poput brzine, pozdanosti, robusnosti, sigurnosti itd.

Sada već možete prepoznati koje sve nivoe testiranja imamo savladane u dosadašnjem radu. Prvi nivo ili nivo testiranja jedinica smo ostvarili primenom unittest i pytest biblioteka Pythona, gde smo direktno u kodu postavljali funkcije koje obavljaju provere rezultata rada jedinica.

Zatim, drugi nivo – nivo integracije – realizovali smo pomoću Jenkinsa. Dakle, test servera koji je naše fajlove spajao na jednoj lokaciji, gde smo pokrenuli testove koji prate rad između više celina našeg koda. Integraciono testiranje može biti i white box i black box vid testiranja, jer integracioni test može napisati menadžer projekta koji ne mora nužno znati svaki deo koda, već samo očekvani rezultat, a ujedno, integracioni test je white box kada ga piše sam programer koji je radio na više modula programa.

Treći nivo, nivo testiranja sistema, nam upravo sledi u narednim lekcijama, ali pre toga, važno je upoznati se sa različitim white box i black box pristupima testiranju koji obuhvataju treći nivo testiranja, ali i generalno dobre pristupe testiranju aplikacija.

White box tehnike testiranja

Generalno, tehnike white box testiranja mogu sadržati sledeće vidove testova:

1. Unit testing
2. Integration testing
3. White box penetration testing
4. White box mutation testing

Naravno, odmah napomenimo da su osnove tehnike ovog vida testiranja unit testovi i testovi integracije. Kako smo ovim pristupima već posvetili neke prethodne lekcije, sada nećemo govoriti o njima, već o drugim pristupima white box testiranja. Takođe je važno reći da su unit testovi i integracioni testovi ujedno i jedini testovi za koje postoje šabloni kako se test piše i gde se i konkretno pišu testovi. Dva sledeća vida testiranja nemaju te šablone, već se oslanjaju na iskustvo programera ili testera koji treba da otkrije probleme.

Testiranje penetracije

White box testiranje penetracije je jedini vid testiranja bele kutije koji se tiče bezbednosti aplikacije. U okviru ovog vida testiranja, tester ili programer ima sve informacije o načinu rada programa i ima pristup izvornom kodu. Pored ovoga, ima potpuni pristup mrežnim informacijama, poput IP adresa, a takođe i informacije o serveru i pristupu serveru. Čitav cilj testa jeste da napada aplikaciju i na taj način otkrije bezbednosne propuste.

Osoba koja obavlja ovaj vid testiranja, pored osnovnog poznavanja jezika u kojem je program napisan, mora znati i generalne načine i metode kako dobiti nedozvoljen pristup aplikaciji. Ovo iziskuje odlično poznavanje funkcionisanja interneta, mrežnih protokola i mreža generalno.

Kao što smo i ranije rekli, ovde ne postoje šabloni, već se pregledaju različiti delovi programa u potrazi za manama. Ove mane mogu biti npr. loše napisana validacija na formama, propusti prilikom registracije korisnika, loše napisana logika prosleđivanja podatka u bazu, pa čak i do sitnica poput otvorenog porta aplikacije koji se nikada ne zatvara.

Mutaciono testiranje

Mutaciono testiranje se koristi kod dizajna novih testova ili provere kvaliteta postojećih testova. Mutant je manje modifikovana verzija programa pod testom, na osnovu malih grešaka. Svaki test slučaj testira i original i sve mutante programa i posmatra se uspešnost test slučaja u identifikovanju razlike između programa i mutanta. Žargonski se ovo zove „killing the mutant”. Spada pod evaluaciju kvaliteta softverskog testa.

U okviru ovog pristupa testiranju koriste se unit testovi i najčešće su to samo varijacije osnovih testova koji su obavljeni, ali sa blago izmenjenim uslovima (mutantima). Dobro napisan unit test obuhvata sve uslove i stoga nema prostora za mutante. Npr. originalni unit test je napisan tako da proverava da li je rezultat veći od nule i ako se kao rezultat dobije broj veći od nula – super. Ali šta ako aplikaciji ne odgovara npr. broj 0.1235, već joj je potreban ceo broj ili čak broj sa maksimalno dve decimale? Ovi slučajevi bi bili mutant slučajevi. Osnovni uslov je da je broj pozitivan, ali na kraju, taj pozitivan broj ima i neke druge, dodatne zahteve. Sada možemo da dodatnim unit testovima omogućimo usklađivanje rada programa ili da sa dodatnim testovima ispratimo rad programa.

Kao i u prethodnom slučaju, ove testove obavljaju softver testeri specijalizovani za proveru već napisanih testova, što već postaje deo oblasti software testing.

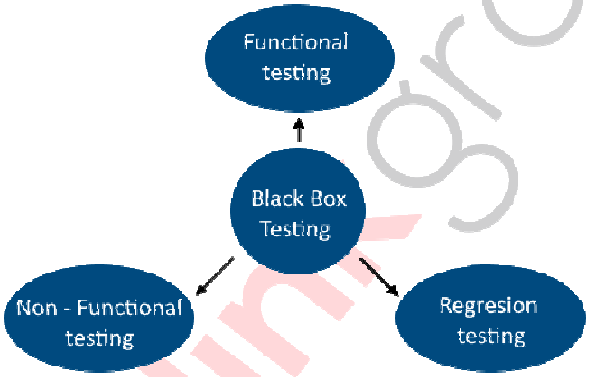
Black box tehnike testiranja

U okviru black box testiranja postoji veliki broj tipova testova. Svi testovi se mogu postaviti u tri osnovne kategorije:

Funkcionalno testiranje – Ovaj tip testiranja crne kutije je fokusiran na funkcionalne zahteve završenog programa/sistema. Testiranje obavlja sofver tester.

Nefunkcionalno testiranje – Kod ovog tipa testiranja fokus nije direktno na konkretnim funkcionalnostima programa, već na performansama, skalabilnosti i upotrebljivosti.

Regresiono testiranje – Regresiono testiranje predstavlja ponavljanje obavljenih testova kada se dogodi promena verzije softvera ili ispravke koda ili nakon redovnog održavanja.



Slika 10.2. Tipovi black box testiranja

Funkcionalno testiranje

Funkcionalno testiranje u relevantnoj literaturi često možemo pronaći pod nazivima *functional testing*, *conformance testing* i *correctness testing*.

Cilj funkcionalnog testiranja je da se ustanovi da li softver odgovara svojim zahtevima. Funkcionalno testiranje se fokusira na unošenje određenih ulaznih parametara u sistem i verifikaciju izlaznih podataka i stanja. Sprovodi se najčešće tokom sistemskog testiranja i koncept funkcionalnog testiranja je prilično sličan za sve sisteme, iako se ulazi i izlazi razlikuju u zavisnosti od tipa i kompleksnosti sistema.

Funkcionalno testiranje daje odgovor na pitanje tipa: „Da li ova funkcionalnost radi?"

Pored testiranja korišćenjem programa i praćenjem ponašanja, jedan od specifičnih pristupa funkcionalnom testiranju je i usability testiranje. U okviru usability testiranja, vrši se evaluacija koliko je jednostavno za krajnje korisnike da savladaju i koriste softver, uključujući i korisničku dokumentaciju.

Testiranje je zasnovano na realnim korisnicima i njihovoj upotrebi softvera. Ovaj vid testiranja i treba da odgovori na pitanje: „Koliko su efikasne softverske funkcije u podršci korisničkim zadacima?" Najčešće se odnosi na web sajtove i aplikacije, korisničke interfejse i sl. – dakle, na sve programe gde korisnici ili organizacija konstantno obavljaju unos, iščitavanje i korišćenje interfejsa.

Ciljevi ovakvog testiranja su utvrđivanje:

- efikasnosti – koliko koraka je potrebno za kompletiranje zadatka, npr. online shop;
- tačnosti – koliko grešaka čine korisnici i koliko su one kritične;
- pamćenja – koliko korisnici pamte nakon perioda nekorišćenja;
- emotivnog utisak – kako se korisnik oseća nakon rada, da li je zadovoljan, pod stresom, da li bi softver preporučio nekome i sl.

Jedna vrsta ovog testiranja je comparative usability testing (primer: poređenje korišćenja jednog sajta sa drugim sličnim). Potreba za upoređivanjem se javila od samih klijenata, gde pristup često bude: želim aplikaciju kao što moja konkurencija ima. Tokom svih faza razvoja, naručilac vrši poređenje; i na samom kraju, vrši se poređenje potrebnih sličnosti, bilo sa aspekta interfejsa ili alata dostupnih unutar programa.

Pitanje

Izmenjenja verzija programa kojom se proverava ispravnost testova nosi naziv:

- **mutant**
- kopija
- virus

Objašnjenje:

Mutant je modifikovana verzija programa pod testom.

Nefunkcionalno testiranje

Nefunkcionalno testiranje predstavlja testiranje kvalitativnih aspekata softvera. Kvalitativni aspekti softvera koji su sami po sebi nefunkcionalni jesu: bezbednost, stabilnost, performanse i pouzdanost. Dakle, u fokusu ovog vida testiranja jeste precizno utvrđivanje načina na koji ceo sistem radi u odnosu na njegovo ponašanje. Ovo je takođe važan aspekt, jer utiče na zadovoljstvo korisnika prilikom korišćenja programa.

Pod nefunkcionalnim testiranjem u osnovi obavljamo stres i load testiranje, testiranje performansi i testiranje prihvatljivosti.

Testiranje performansi softvera

Testiranje performansi softvera ima za cilj da ustanovi da li softver radi ispravno u odnosu na zahteve u vezi sa brzinom i vremenom izvršavanja određenih zahteva, pošto je utvrđeno da sistem izvršava funkcije navedene u zahtevima – naravno, funkcionalnim testiranjem. Sada se prelazi na način na koji se te funkcije izvršavaju i ovo je jedan od prvih testova nakon funkcionalnih testova.

Testiranje performansi ispituje kvalitet izvršavanja. Brzina odgovora na komande korisnika, preciznost rezultata i dostupnost podataka porede se sa definisanom performansom, po mogućstvu izraženom kvantititavno.

Stres i load testiranje

Stres i load testiranje predstavlja deo testiranja performansi. U okviru ovog testiranja se vrši ispitivanje sistema pod velikim opterećenjem. Podrazumeva test stabilnosti sistema u situacijama izvan normalnog operativnog kapaciteta i tada ima za cilj da se provere performanse sistema u ekstremnim uslovima rada i da se na taj način utvrde granice izdržljivosti sistema. Često se sprovodi sa izuzetno velikim brojem korisnika i sa ogromnom količinom podataka.

Na ovaj način se mogu proveriti opterećenja kod velikog broja posetilaca na nekom web serveru ili napraviti simulacije napada uskraćivanjem usluge (DOS napadima) preko raznih skripti, web robota i dr. U narednim lekcijama, stres i load testiranje obavljaćemo pomoću softvera koji je industrijski standard u ovoj oblasti. To je aplikacija JMeter.

Testiranje prihvatljivosti

Ova vrsta testiranja često se u relevantnoj literaturi navodi kao *acceptance testing*. Testiranje prihvatljivosti ispituje da li se sistem ponaša ispravno kroz duži vremenski period. Kada se završi funkcionalno testiranje i testiranje performansi, naručilac preuzima ključnu ulogu, vodi ovaj tip testiranja i definiše slučajeve koji će se testirati.

Svrha testa prihvatanja jeste da se kupcima i korisnicima omogući da utvrde da li sistem zaista zadovoljava njihove potrebe i očekivanja. Sastav tima i vrstu testova definiše naručilac; tim može, ali i ne mora da uključuje članove razvojnog tima.

Testiranje se ne sprovodi po striktnoj proceduri i testovima, već je više *ad hoc* tipa (bez plana i dokumentacije), gde korisnici jednostavno vrše interakciju sa softverom i prijavljuju svoja zapažanja.

Može se izvršiti interni i eksterni test prihvatanja:

- ***interni*** – alfa testiranje, gde testeri i potencijalni korisnici sistema testiraju proizvod pod pretpostavkama koje je definisala organizacija;
- ***eksterni*** – beta testiranje; posle uklanjanja grešaka pronađenih u alfa testiranju, sprovodi se beta testiranje, gde korisnici testiraju program i izveštavaju o pronađenim greškama.

Nakon svakog testiranja prihvatljivosti, naručilac obaveštava koji zahtevi nisu zadovoljeni. alpha i beta testiranje se koriste za aplikacije koje su predviđene za veliki broj korisnika, jer je tada teže uočiti problem bez upotrebe programa u realnim situacijama.

Regresiono testiranje

Regresiono testiranje je selektivno retestiranje sistema ili komponenata radi verifikacije da modifikacije nisu prouzrokovale neželjene efekte i da se nisu ponovo pojavile stare greške. Regresivni testovi su testovi koji proveravaju da li funkcionalnosti koje su radile u prethodnoj verziji rade i u novoj verziji. Ako ne rade, to se naziva regresija. Cilj je osigurati da promene (unapređenja, zakrpe i sl.) nisu prouzrokovale greške.

Regresionim testiranjem se takođe utvrđuje da li je promena u jednom delu softvera uticala na druge delove softvera. Ono što je specifično za ovaj vid testiranja je to što ono može biti okarakterisano i kao i white box i kao black box testiranje. Razlog je to što se ovde proveravaju svi aspekti softvera, dakle od jednostavnog unit testa pa do promena npr. u brzini izvršavanja aplikacije do kojih dolazi mnogo kasnije u razvoju aplikacije. Stoga regresiono testiranje može da se sprovede na svakom test nivou.

Najčešći metod regresionog testiranja podrazumeva ponovno pokretanje ranijih testova i proveru da li su se prethodno ispravljene greške ponovo pojavile, što bi bio prvi indikator da je promena izazvala problem u radu softvera.

Sada smo se upoznali sa važnom teorijskom osnovom iza funkcionalnih i nefunkcionalnih testova viših nivoa; u narednim lekcijama fokusiraćemo se upravo na načine njihove realizacije.

Rezime

- U okviru whitebox pristupa testiranju možemo uraditi sledeće testove: unit test, integration test, penetration test, mutation test.
- White box testiranje penetracije je vid testiranja bele kutije koji se tiče bezbednosti aplikacije. U okviru ovog vida testiranja, tester ili programer ima sve informacije o načinu rada programa, pristup izvornom kodu, kao i pristup mrežnim informacijama i informacijama o serveru. Čitav cilj testa jeste otkrivanje načina za napad aplikacije i, na taj način, otkrivanje bezbednosnih propusta.
- Mutaciono testiranje se koristi kod dizajna novih testova ili provere kvaliteta postojećih testova. U okviru ovog pristupa koriste se unit testovi i najčešće su to samo varijacije osnovih testova koji su obavljeni, ali sa blago izmenjenim uslovima (mutantima). Dobro napisan unit test obuhvata sve uslove i stoga nema prostora za mutante.
- U okviru black box testiranja postoji veliki broj tipova testova. Svi testovi se mogu postaviti u tri osnovne kategorije: funkcionalno testiranje, nefunkcionalno testiranje i regresiono testiranje.
- U okviru usability testiranja, vrši se evaluacija koliko je jednostavno za krajnje korisnike da savladaju i koriste softver, uključujući i korisničku dokumentaciju.
- Stres i load testiranje predstavlja deo testiranja performansi. U okviru testiranja se vrši ispitivanje sistema pod velikim opterećenjem. Podrazumeva test stabilnosti sistema u situacijama izvan normalnog operativnog kapaciteta kako bi se proverile performanse sistema u ekstremnim uslovima rada i utvrdile granice izdržljivosti sistema.
- Testiranje prihvatljivosti ispituje da li se sistem ponaša ispravno kroz duži vremenski period. Može da se podeli na interni (alpha test) i eksterni test (beta test) prihvatanja.
- Regresiono testiranje je selektivno retestiranje sistema ili komponenata radi potvrde da modifikacije koda nisu prouzrokovale neželjene efekte. Regresivni testovi su testovi koji proveravaju da li funkcionalnosti koje su radile u prethodnoj verziji rade i u novoj verziji. Ako ne rade, to se naziva regresija. Cilj je osigurati da promene (unapređenja, zakrpe i sl.) nisu prouzrokovale greške.